Dennis Ward

Project One

4/12/2024

BigO Analysis

Vector runtimes





The total cost to use a vector is 14n+3 simplified to BigO(n). The advantages with a vector is there is not a great fluctuation in runtime efficiency. It is simple and easy to follow code with low maintenance.

Hashtable Runtimes





The total cost to use a vector is 13n+6 simplified to BigO(n). A Hashtable can have excellent run time operations if collision is low. Although with this data set it will likely have high collision unless the hash key forming process is excellent. The items are not stored in a sorted order making it more difficult for printing process that which to be in alphanumeric order.

BST Runtime





Total cost is (3n+log(n)\*n + 3 ) + (3n+3nlog(n)) = 6n+3+4nlog(n) Runtime is BigO(nlog(n))

A BST has excellent run time operations if the BST is balanced. This data is perfect for a BST as it is unsorted, meaning it will lead to a well-balanced BST. Deleting operations are the main concern to cause an unbalanced BST. Deletion will happen if all courses are not valid and could cause imbalance. BST’s need to be well maintained in order to be efficient.

In after doing the analysis I’m a little shaky on the meanings of it all but as the data grows the Hashtable becomes the best-case runtime. Because while its worst cast is BigO(n) it is likely that some operations will be BigO(1) making it slightly better than a vector. While I also believe that this is not the right choice for the use case. The most operations needed is printing the HashTable in alphanumerical order. Which is only possible by using the full course ID as the key which comprises turning the letters into numbers, which would still have extreme collision because of the similarity of the numbers or loading the hash table into another struct type and sorting it and printing it.

After all of what is said I believe the best runtime only considering the initial load and validation would be the Hashtable. If you were to consider, any kind of long-term usage I think it would be better to Load the data into a vector and validate and then move it to a BST for operational usage. This would lead to a well-balanced BST, which means quick operational runtime BigO(log(n)) on instructions while in use with the advisors. The data is also sorted, making print functions quicker. It depends more on which operation needs to be the most efficient. There is an argument for using a hashtable instead of a vector but a hashtable is semi sorted making it bad to load into a BST.

Therefor my recommendation is to use a vector, due to it being better run time than a BST and only slightly less than a HashTable.